MANUAL DE CLOCK PARA PICS

Por: Thomas Henry Tradução, Inserções e Notas Diversas: Herbert Lopes

---

Todos os microcontroladores requerem um relógio ou oscilador (vulgarmente chamado de “clock”) para guiar um programa através de seus passos. É dever de tal módulo indicar quando uma instrução deve ser buscada da memória do programa, decodificada e executada. Na verdade, mesmo a instrução mais simples é composta de uma série de operações que devem ser sequenciadas na ordem certa e no momento certo. O relógio, então, é como o maestro de uma orquestra, coordenando todas as partes que compõem o todo.

Como o relógio é tão importante, os microcontroladores PIC oferecem uma ampla gama de opções para escolher o melhor para suas próprias aplicações. Você pode ter relógios que funcionam rapidamente quando muita coisa tem que acontecer em um curto espaço de tempo, ou aqueles que consomem energia insignificante para equipamentos operados por bateria. Talvez sincronizar o PIC com um relógio de tempo real estável seja importante para você. Na verdade, há oito ou mais opções de relógio para o PIC típico.

No entanto, se você pegar a folha de dados (ou datasheet, se preferirem) e tentar classificar todas as possibilidades, você rapidamente se arrependerá daquela velha frase, "Não consigo ver a floresta por causa das árvores". Datasheets para PICs têm centenas de páginas e não são necessariamente organizadas para o melhor aprendizado (sendo bem benevolente com os engenheiros que os fazem). E, claro, o erro ocasional aparece, tornando ainda mais difícil para o novato entender o que está acontecendo e se atualizar.

Como alguém que passou toda a sua vida adulta ensinando, sempre valorizei intuitivamente a importância de uma apresentação bem organizada. Você começa com a "floresta" e só aborda as "árvores" depois. É isso que faremos aqui — tendo o panorama geral em mente primeiro e, então, abordando os detalhes quando eles finalmente forem necessários.

Então, se você ficou consternado no passado com o quão complicadas as opções de clocks para PICs parecem ser, venha agora e veja como a abordagem certa pode fazer toda a diferença. Para transformar isso em uma experiência de aprendizado ativa, concluiremos com alguns experimentos reais que você pode conduzir na própria protoboard (no Proteus, funciona tudo mais fácil que na vida real, a coisa física tem mais tendência a dar problemas em grande parte dos casos). Ao final da nossa sessão juntos, você deve estar pronto para começar a usar clocks em microcontroladores PIC com confiança.

CLOCKS PRIMÁRIOS

Para manter as coisas concretas, vou me concentrar no PIC16F88, que é um dos microcontroladores mais populares entre os DIYers. No entanto, outros PICs ostentarão muitas das mesmas opções — até mesmo os chips menores de seis e oito pinos das famílias 10 e 12, embora a família 10 apresente menos opções.

O primeiro conceito a ser definido é que o PIC pode ser executado em um clock primário ou secundário. O primeiro é responsável pela maioria das situações. Em particular, um projeto típico contém um único clock guiando o microcontrolador e esse é o fim da questão. Em aplicações mais avançadas, podemos querer que um clock secundário entre em ação sempre que o clock primário estiver fora de cena. Existem vários motivos para desejar tal redundância, e vamos explorá-los em breve.

Por enquanto, vamos nos concentrar no clock primário, que é tudo o que precisamos para a maioria dos projetos. Consulte a Figura 1, que mostra o que está disponível em tecnologias de clock primário.

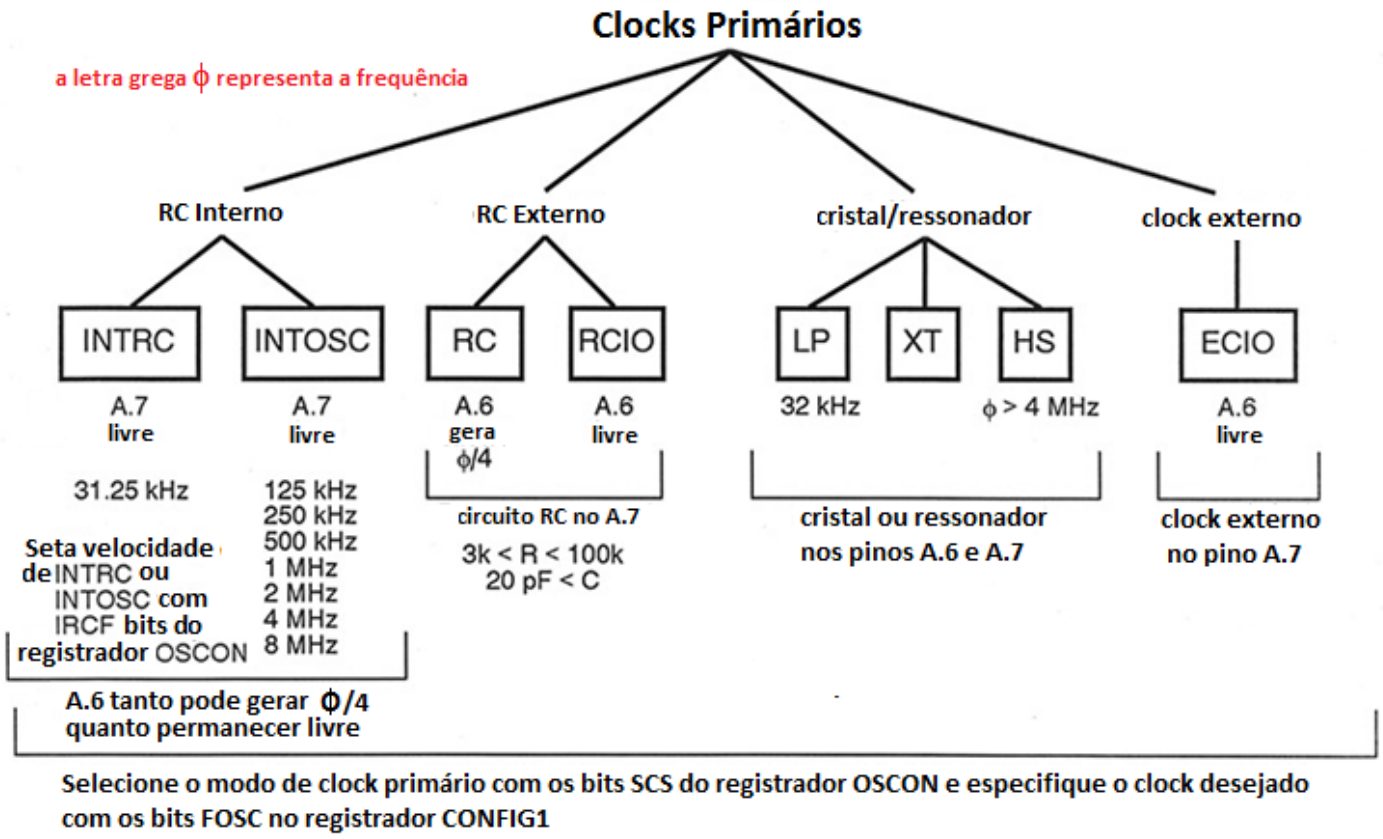

FIGURA 1. Os oito modos de clock primário.

Este diagrama de árvore organiza os tipos de clocks primários disponíveis no PIC16F88 de cima para baixo. Para começar, observe os primeiros ramos do diagrama de árvore. Um relógio pode ser um RC interno ou, se preferir, você pode fornecer seu próprio resistor e capacitor e terminar com um arranjo RC externo. Para situações que exigem mais precisão, um ressonador externo de cristal ou cerâmica pode ser empregado. Por fim, se desejar, você pode acionar o PIC com um clock externo gerado (CI LMC55, por exemplo).

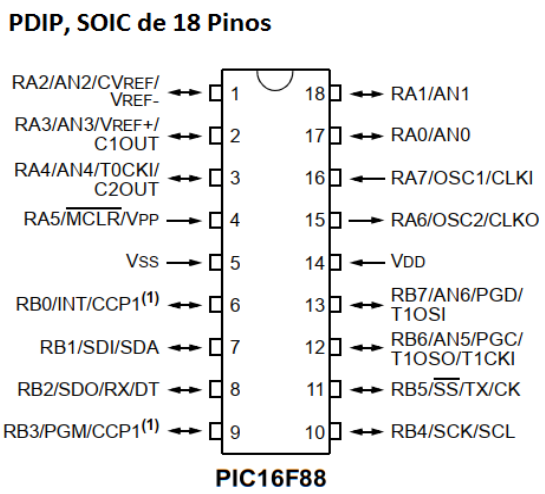

FIGURA 2

Adicionando um pouco mais de detalhes agora, os ramos da árvore se dividem novamente, mostrando os oito modos individuais. Cada um tem um nome abreviado dado a ele pela Microchip. Vale a pena aprender agora para simplificar a leitura do Datasheet mais tarde. Vamos escanear esses modos específicos da esquerda para a direita. O primeiro é INTRC, que significa oscilador RC interno. Rodando a aproximadamente 31 kHz, este é um caso de baixa velocidade construído dentro do PIC. Embora seja bem lento para qualquer padrão, tem a vantagem de ser simples e consumir muito pouca corrente. Infelizmente, esta modalidade de clock não está presente em muitos PICs, como a família 16F7X. A família citada (16F7X) só tem modo de RC externo, onde você configura um capacitor em VDD e um resistor em VSS ligados ao pino 9 do chip (OSC1).

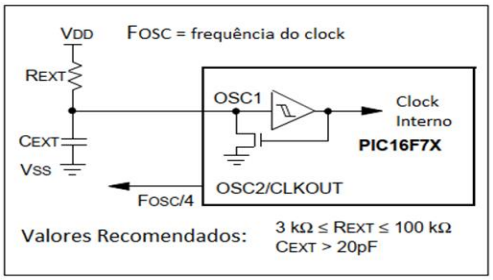

FIGURA 3

Nas experiências de laboratório algumas combi-nações de Rext e Cext simplesmente não funcio-naram. Os componentes estavam bons, foram ligados corretamente mas o clock do microcontrolador não oscilava de forma alguma. A maioria porém funcionou satisfatoriamente, conforme previsto.

O próximo modo é INTOSC. Como INTRC, este é interno (não requer componentes externos – e igualmente não existe na família PIC16F7X), mas pode rodar em sete velocidades diferentes até 8 MHz. Em alguns dos PICs, INTRC e INTOSC são realmente independentes, mas não no PIC16F88 que estamos examinando. Por esta razão, no Datsheet os escritores às vezes usam INTRC para significar apenas INTRC e outras vezes para significar INTRC ou INTOSC. Isso só aumenta a confusão, então vou manter os nomes separados.

Em qualquer um desses modos, a linha de porta A.6 pode ser configurada para seguir o clock dividido por quatro, se desejado. Isso é simbolizado por φ/4 — leia "phi dividido por quatro". Na família PIC16F7X teremos o pino 10 (OSC2/CLKOUT) desempenhando a mesma função. Use essa saída se quiser sincronizar algum equipamento externo com o micro. O Datasheet chama esse modo de INTIO1, no caso do PIC16F88, e ele não existe na família PIC16F7X. Caso contrário, você pode opcionalmente liberar o pino A.6 para E/S digital comum, chamado como modo INTIO2, que igualmente não existe na PIC16F7X.

Passando para os modos RC externos, o primeiro é simplesmente chamado de RC. Nesse caso, um resistor externo e um capacitor no pino A.7 (OSC1/CLKIN em outros PICs) fazem as coisas funcionarem. Assumindo uma fonte de alimentação de +5 V, o resistor deve ficar entre 3 K e 100 K, enquanto o limite deve ser maior que 20 pF. Com os menores valores, o clock atingirá o máximo em torno de 4 MHz. Como vimos com os clocks internos — se desejado — A.6 pode ser feito para seguir φ/4 (esse é o modo RC) ou então ficar livre para uso comum (esse é o RCIO). A propósito, o resistor poderia ser um potenciômetro conectado como um reostato, o que lhe daria um clock variável, mas para a imensa maioria das aplicações utilizamos clocks fixos – sendo que em algumas mais específicas, variações no clock prejudicam o funcionamento do projeto.

Cristais e ressonadores cerâmicos são atraentes, pois são muito mais precisos tanto a curto quanto a longo prazo, e também com relação a tolerâncias e temperatura. Existem três modos aqui. O primeiro, LP — que significa baixa potência — deve ser usado com um cristal tipo diapasão de 32,768 kHz, dos mais populares do mundo por serem utilizados em relógios diversos (parede, pulso, etc) e em clocks de tempo real, como os que controlam a data e hora em PCs. Cristal e capacitores de carga associados se conectam aos pinos A.6 e A.7.

O próximo modo — chamado XT — é semelhante, mas destinado ao uso com cristais e ressonadores de frequência mais alta até cerca de 4 MHz. O último modo é HS — que significa alta velocidade — e é necessário para cristais ou ressonadores que giram acima de 4 MHz. Só para você saber, o limite superior para o PIC16F88 é 20 MHz. O interessante sobre esses três modos é que o chip seleciona automaticamente o ganho correto do driver interno para o cristal com base na especificação de LP, XT ou HS. O último modo é o ECIO, que simplesmente permite que você injete um sinal de clock externo no pino A.7. Isso pode vir de qualquer tipo de circuito de clock que emita pulsos quadrados (ou bem próximos de quadrados) oscilando do terra para a tensão de alimentação.
Dando uma espiada nos experimentos, a Figura 6 mostra os arranjos de hardware envolvidos. Veremos como estipular qualquer um desses oito modos em apenas um momento, mas primeiro vamos dar uma olhada nas opções de clock secundário.

CLOCKS SECUNDÁRIOS (DIVERSOS PICS NÃO TEM)

A Figura 4 mostra um diagrama de árvore de ramificação para os clocks secundários disponíveis. Lembre-se de que qualquer um deles pode assumir funções do relógio primário sob certas circunstâncias. A planilha do Datasheet gosta de agrupá-los nas categorias RC_RUN e SEC_RUN, mas ambos indicam relógio secundário.

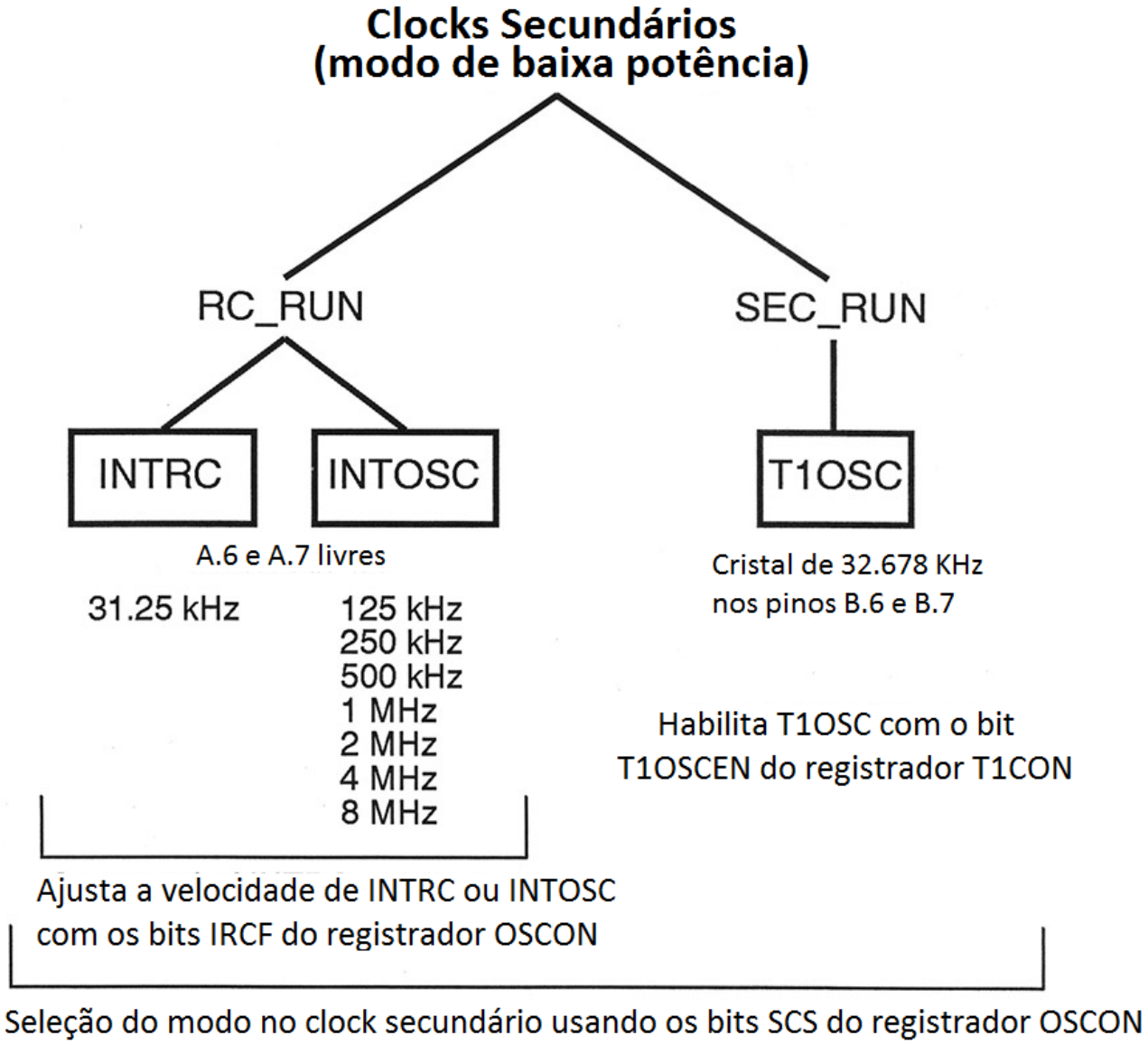

FIGURA 4. Os três modos para clocks secundários.

Em linhas gerais, temos que modos de baixo consumo: RC_RUN permite que o núcleo e os periféricos sejam sincronizados a partir do INTRC, enquanto SEC_RUN permite que o núcleo e os periféricos sejam sincronizados a partir do Timer1 de baixo consumo. Consulte o Datasheet do PIC16F88, na **Seção 4.7** “Modos de gerenciamento de energia” para mais detalhes.

Há duas opções para RC_RUN. Os relógios internos INTRC ou INTOSC descritos anteriormente podem ser usados; a única diferença é realmente a taxa de oscilação.

Ou, se desejar, você pode conectar o cristal de 32,768 kHz e os capacitores de carga a B.6 e B.7. O que é tão legal sobre isso é que o cristal do relógio sempre funciona — mesmo quando você coloca o chip no que é conhecido como modo de espera.

Este modo — designado T1OSC — é ideal para aplicações de relógio em tempo real.

(Normalmente, o PIC desligará os relógios não utilizados ao entrar em espera).

Observe que os relógios primários usam A.6 e A.7 em várias combinações, mas os relógios secundários não — pelo menos para o PIC16F88. O esquema na Figura 7 (para um dos experimentos que estão por vir) mostra o que está envolvido.

Com isso, concluímos nossa visão geral dos relógios primários e secundários. Por que não reservar alguns minutos para estudar as Figuras 1 e 2 mais uma vez para realmente fixar as distinções e os detalhes em sua mente antes de prosseguir?

COMO FAZER SUA ESCOLHA:

Então, você decidiu que tipo de relógio quer. Agora, como você expressa seus desejos para o PIC? A resposta está nos bits de configuração e três registradores especiais. Os bits de configuração são definidos durante a fase de gravação (Flashing ou programação), enquanto os bits dos registradores são acessíveis durante o tempo de execução. Aqui está o grande lance! Lembre-se de sempre verificar o Datasheet do PIC.

A **Figura 5** ilustra os dois conjuntos de bits de configuração dentro do PIC16F88. (PICs mais simples têm apenas um conjunto.) Sim, há muito conteúdo a ser aprendido aqui, mas quando se trata de configurar o clock, apenas um punhado básico precisa nos preocupar. Por exemplo, os três bits rotulados FOSC (no PIC16F88) e que podem ser apenas dois (no PIC16F76) designam o oscilador primário, conforme descrito anteriormente. Em outras palavras, eles estabelecem o comportamento padrão na inicialização.

REGISTRADOR CONFIG1

CP	CCPMX	DEBUG	WRT1	WRT0	CPD	LVP
bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7

BOREN	MCLRE	FOSC2	$\overline{\text{PWRTEN}}$	WDTEN	FOSC1	FOSC0
bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

FOSC2 : FOSC1 : FOSC0

Os bits do registrador FOSC são usados para setar o modo do oscilador primário

111 RC
110 RCIO
101 INTRC or I
100 INTRC or I
011 ECIO
010 HS
001 XT
000 LP

REGISTRADOR CONFIG2

n/a	n/a	n/a	n/a	n/a	n/a	n/a
bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7

n/a	n/a	n/a	n/a	n/a	IESO	FCMEN
bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

IESO = Transição Interno/Externo
FCMEN = Habilitação de clock à prova de falhas

IESO é usado para habilitar o modo de inicialização do relógio de duas velocidades. Ele é necessário somente quando o relógio primário é LP, XT ou HS. Após uma reinicialização, o relógio será INTRC brevemente, alternando a fonte primária assim que o cristal tiver aquecido.

Se o bit FCMEN estiver ligado e o relógio falhar em modo LP, XT, HS ou ainda se T1OSC falhar, a operação continuará automaticamente com INTRC ou INTOSC.

FIGURA 5 A configuração dos Bits é feita durante a gravação do PIC

No segundo conjunto de bits de configuração, você encontrará dois que governam o que acontecerá quando você alternar dos relógios primários para os secundários, ou vice-versa. Vamos deixar isso para a próxima seção. Uma vez que um programa esteja em execução, você pode manipular o clock (ou clocks) de várias maneiras, modificando os registradores OSCON, OSCTUNE e T1CON. Eles são mostrados na **Figura 6**. Vamos ponderar os detalhes.

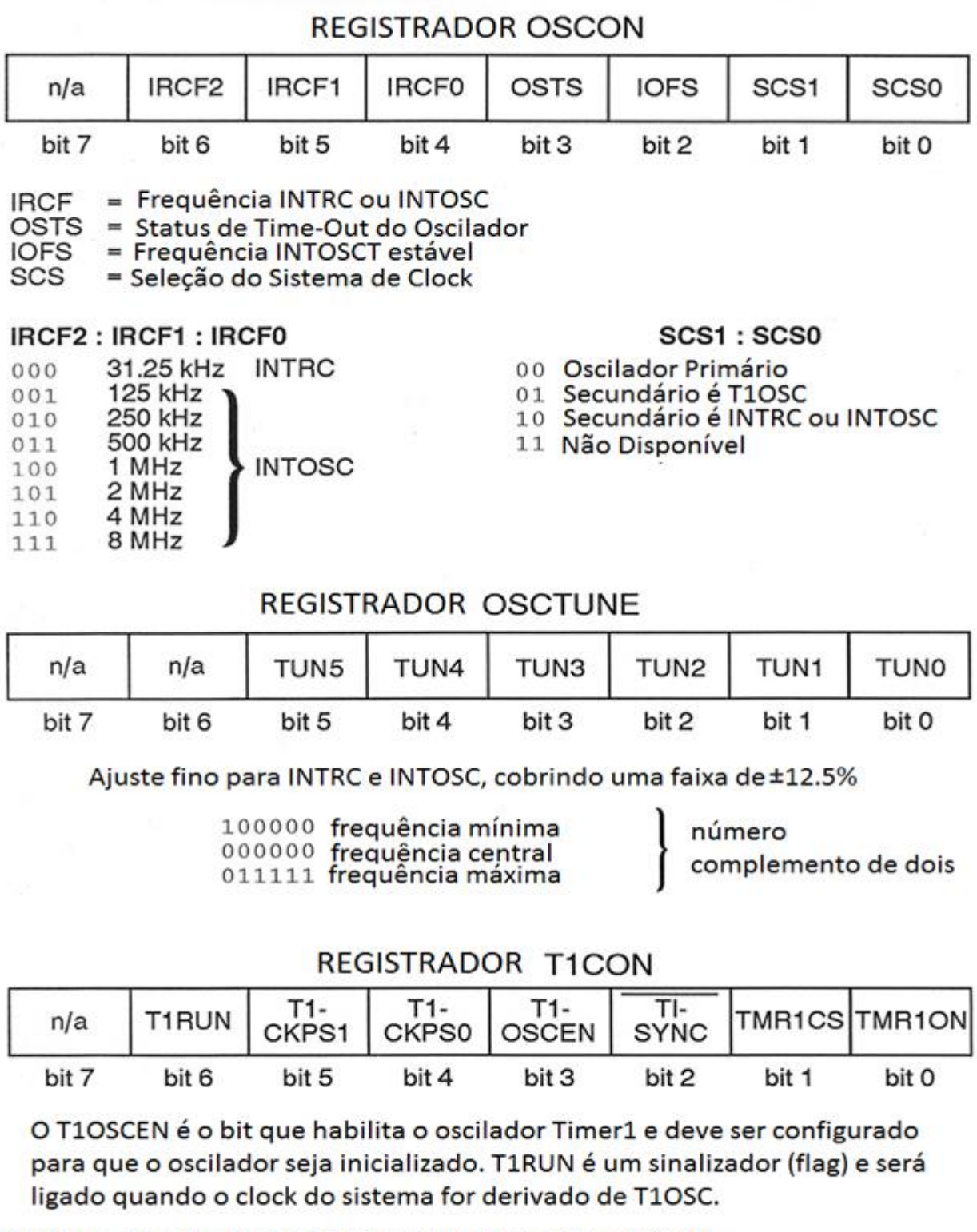
REGISTRADOR OSCON
n/a IRCF2 IRCF1 IRCF0 OSTS IOFS SCS1 SCS0
bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
IRCF = Frequência INTRC ou INTOSC
OSTS = Status de Time-Out do Oscilador
IOFS = Frequência INTOSCT estável
SCS = Seleção do Sistema de Clock
IRCF2 : IRCF1 : IRCF0
000 31.25 kHz INTRC
001 125 kHz
010 250 kHz
011 500 kHz
100 1 MHz INTOSC
101 2 MHz
110 4 MHz
111 8 MHz
SCS1 : SCS0
00 Oscilador Primário
01 Secundário é T1OSC
10 Secundário é INTRC ou INTOSC
11 Não Disponível
REGISTRADOR OSCTUNE
n/a n/a TUN5 TUN4 TUN3 TUN2 TUN1 TUN0
bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
Ajuste fino para INTRC e INTOSC, cobrindo uma faixa de ±12.5%
100000 frequência mínima
000000 frequência central
011111 frequência máxima
número
complemento de dois
REGISTRADOR T1CON
n/a T1RUN T1-CKPS1 T1-CKPS0 T1-OSCEN TI-SYNC TMR1CS TMR1ON
bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0
O T1OSCEN é o bit que habilita o oscilador Timer1 e deve ser configurado para que o oscilador seja inicializado. T1RUN é um sinalizador (flag) e será ligado quando o clock do sistema for derivado de T1OSC.

FIGURA 6. SELEÇÃO DE CLOCKS SECUNDÁRIOS DE BAIXA POTÊNCIA

Observe os quadros comparativos para 3 famílias diferentes de PICs e suas arquiteturas:

PIC16F87/88: Bits usados: FOSC2:FOSC1:FOSC0

111 Oscilador EXTRC função CLK0 no pino RA6/OSC2/CLK0	110 Oscilador EXTRC função Porta E/S no pino RA6/OSC2/CLK0	101 Oscilador INTRC funçao CLK0 no pino RA6/OSC2/CLK0 e função de porta E/S no pino RA7/OSC1/CLKI	100 Oscilador INTRC função de porta E/S nos pinos RA6/OSC2 e RA7/OSC1
011 ECIO função de porta E/S no pino RA6/OSC2/CLK0	010 Oscilador HS	001 Oscilador XT	000 Oscilador LP

PIC16F73/74/75/76: Bits usados: FOSC1:FOSC0

11 Oscilador RC	10 Oscilador HS	01 Oscilador XT	00 Oscilador LP

PIC12F629/675: Bits usados: FOSC2:FOSC1:FOSC0

111 Oscil. RC função CLKOUT no pino GP4/OSC2/CLKOUT	110 Oscil.RC função Porta E/S no pino GP4/OSC2/CLKOUT e RC no pino GP5/OSC1/CLKIN	101 Oscil. INTOSC funçao CLKOUT no pino GP4/OSC2/CLKOUT, função E/S no pino GP5/OSC1/CLKIN	100 Oscil. INTOSC porta E/S no pino GP4/OSC2/CLKOUT e porta E/S no pino GP5/OSC1/CLKIN
011 EC função de porta E/S no pino GP4/OSC2/CLKOUT e CLKIN no pino GP5/OSC1/CLKIN	010 Oscilador HS cristal alta velocidade pinos GP4/OSC2/CLKOUT GP5/OSC1/CLKIN	001 Oscilador XT cristal ou ressonador nos pinos GP4/OSC2/CLKOUT GP5/OSC1/CLKIN	000 Oscilador LP cristal de baixa potência nos pinos GP4/OSC2/CLKOUT GP5/OSC1/CLKIN
Talvez os bits mais importantes no OSCON sejam os rotulados IRCF, que significa frequência do oscilador RC interno. Como o nome sugere, esses bits selecionam a taxa de clock desejada do(s) oscilador(es) interno(s). Em seguida, estão os dois bits rotulados SCS, denotando seleção do clock do sistema. Aqui, você pode escolher usar os clocks primário ou secundário. Não é nenhuma surpresa que o registro OSCTUNE permite que você ajuste a frequência do clock INTRC ou INTOSC. (Em alguns PICs, apenas INTOSC é afetado.) Um número de complemento de dois de seis bits é tudo o que é preciso.			

Números negativos desaceleram o clock, enquanto os positivos o aceleram. O ajuste em uma faixa de ±12,5% é possível.

O registro T1CON se preocupa em controlar o Timer 1, que normalmente é sincronizado por um cristal externo de 32,768 kHz, que também pode ter dupla função como clock do sistema. Você pode se lembrar disso como modo T1OSC da Figura 2. O sinalizador importante aqui é T1OSCEN, que habilita o oscilador de cristal Timer 1.

Neste ponto, atingimos os fundamentos do clock do PIC para a maioria das situações comuns. Vamos terminar dando uma breve olhada nos clocks secundários e para que eles servem.

CHAVEAMENTO DO CLOCK

Pode chegar o dia em que você vai querer ir além de apenas ligar um relógio primário e deixá-lo fazer o que tem que fazer. Se for o caso, então você vai precisar saber um pouco sobre troca de relógio. A **Figura 7** dá o panorama geral. Essencialmente, você pode alternar manualmente entre os relógios primário e secundário, ou fazer isso automaticamente sob certas condições.

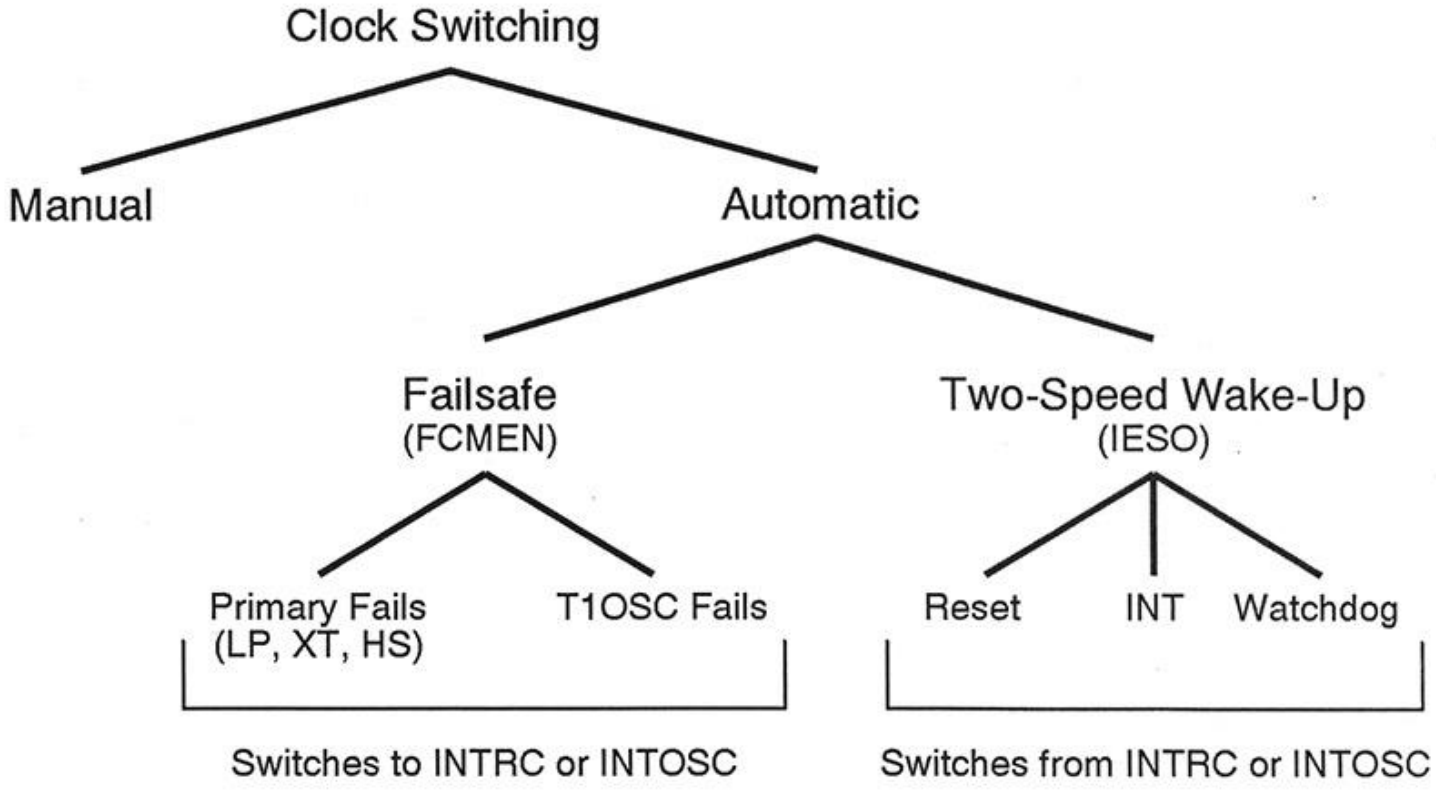

FIGURA 7. Os Clocks podem ser chaveados entre os modos primário e secundário.

Para alternar manualmente de um para outro, você volta para a Figura 4 e simplesmente precisa alterar alguns bits nos três registradores descritos ali. Por exemplo, para alternar de INTRC para INTOSC, você ajustaria os bits IRCF0 a IRCF2 conforme necessário. Ou, para ir de um oscilador de cristal, digamos, para INTOSC, vá para SCS0 e SCS1, que permitem alternar entre o primário e o secundário. Você entendeu a ideia; alternar manualmente os osciladores é apenas uma questão de mexer em bits no seu programa.

Então, e quanto às comutações automáticas? Bem, há duas situações em que é útil ter um relógio substituindo outro. A primeira é o mecanismo de segurança contra falhas. Imagine que você está projetando um circuito para operar em um ambiente crítico em medicina ou monitoramento de segurança industrial, por exemplo. Se o oscilador primário fosse um cristal, por exemplo, e falhasse por algum motivo, o PIC simplesmente pararia. Não é bom! Com um relógio secundário nas asas, o chip poderia alternar para ele quase perfeitamente e ainda continuar operando até que a falha original pudesse ser atendida. Como mostra a Figura 5, o recurso failsafe pode verificar problemas com qualquer um dos quatro modos de cristal (LP, XT, HS ou T1OSC). Ele é habilitado por meio do bit FCMEN no registrador CONFIG2.

Se evitar atrasos for uma preocupação, a opção de despertar de duas velocidades pode ser útil. Aqui está a ideia básica. Suponha que você estivesse operando o circuito em um cristal. Como você provavelmente sabe, os osciladores de cristal levam um ou dois momentos para aquecer. O PIC entende isso e espera pacientemente um pouco antes de deixar o relógio começar a disparar as instruções pelo pipeline. Em outras palavras, seu projeto fica parado ali sem fazer nada por alguns momentos. Por outro lado, os osciladores internos INTRC e INTOSC são essencialmente colocados em marcha instantaneamente. Então, por que não escolher um deles para servir como um relógio secundário?

Agora, a sequência será que o oscilador interno lida com tudo (possivelmente em uma taxa mais lenta, mas pelo menos está se movendo!) até que o cristal esteja pronto para assumir o manto do relógio do sistema. A troca é automática e não há tempo morto onde nada está acontecendo.

Quando isso é necessário? Conforme informado anteriormente, os PICs suportam um comando sleep. Quando executado, o material que consome muita energia é desligado e o chip fica ocioso em um estado de corrente muito baixo. Em particular, qualquer cristal sendo usado como um relógio primário é interrompido. Vários sinais como uma reinicialização, uma interrupção ou algo chamado watchdog timer podem acordar o chip novamente. Se você quiser alguma ação instantânea enquanto o cristal acelera, considere o despertar de duas velocidades que acabamos de descrever. Para habilitar essa resposta, vá para o bit IESO em CONFIG2.

AS EXPERIÊNCIAS

Com isso, cobrimos o essencial da cronometragem do PIC. Há outras sutilezas com as quais você pode se preocupar mais tarde, e é para isso que servem Datasheets e Tabelas. Neste ponto, você pelo menos será capaz de lidar com a maioria das situações comuns e estará em muito melhor forma para realmente ler essa maldita coisa!

Para realmente consolidar o que aprendeu no local, você é encorajado a tentar a dúzia de experimentos descritos na barra lateral abaixo. Você aprenderá literalmente fazendo. As **Figuras 8 e 9** descrevem a configuração do hardware, enquanto o software PIC16F88 é cuidado pelo código-fonte disponível nos arquivos de download no link do artigo. Os programas são escritos na linguagem gratuita de código aberto Great Cow Basic, mas são facilmente portados para PICBasic e outras linguagens. Em cada experimento, o código-fonte inclui muitos comentários para ajudá-lo e também descreve o que fazer para interpretar os resultados. Experimente e veja por si que clockar um PIC não é tão mau.

O arquivo de download deste artigo contém o código-fonte para 13 experimentos usando o microcontrolador PIC16F88 comum e barato. Os exercícios demonstram todos os vários modos de clock. Os programas são bem documentados e contêm instruções sobre como configurar as coisas e o que procurar. O arranjo de hardware para os Experimentos 1 a 12 é descrito na Figura 6, enquanto a Figura 7 mostra o necessário para o Experimento 13.

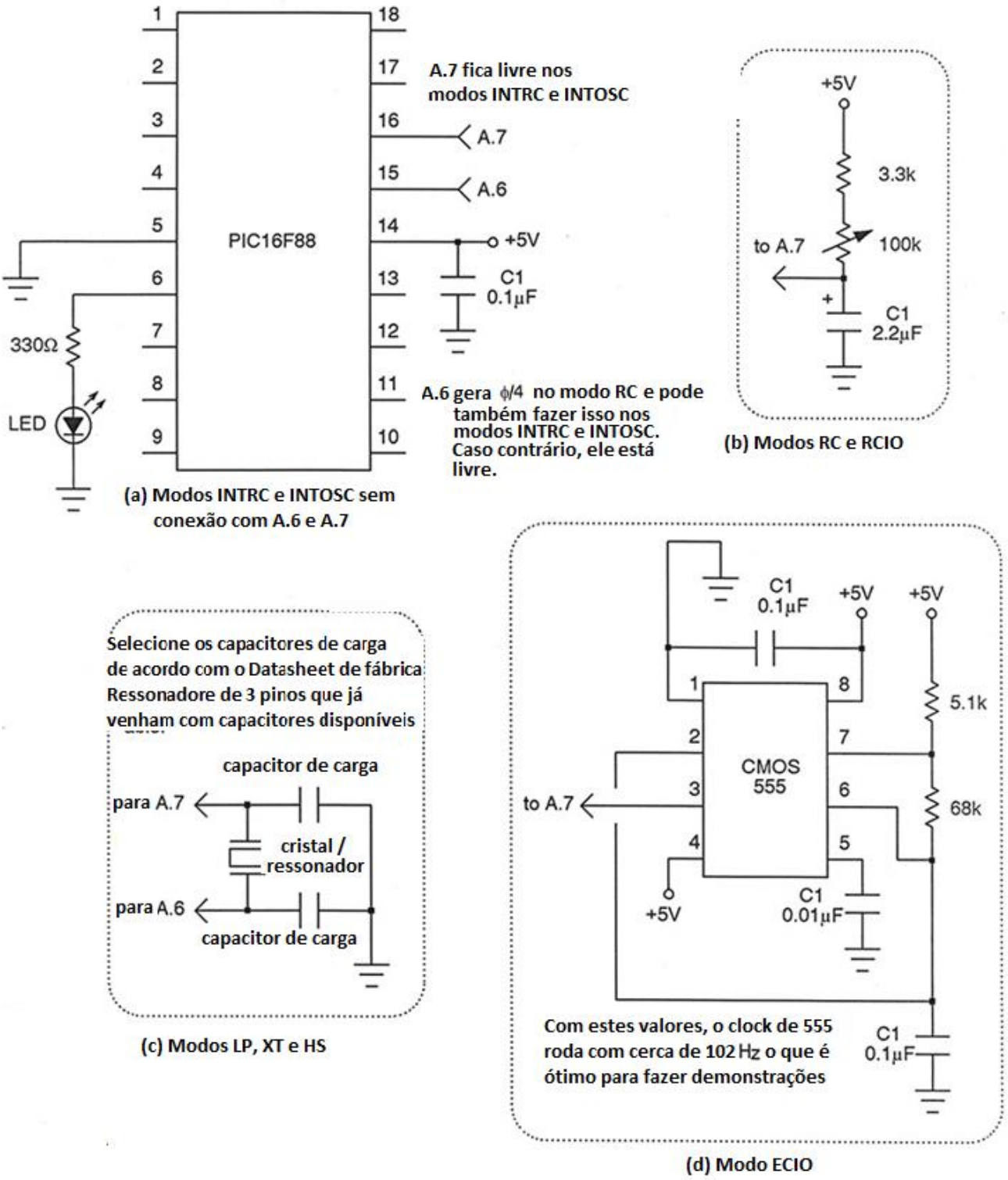

FIGURE 8 . Aqui estão os detalhes de hardware para clocks primários.

Nas experiências 2, 4, e 6, use um osciloscópio ou frequencímetro para monitorar φ/4. E todos os casos, um LED piscará para que o PIC tenha algo pra fazer.

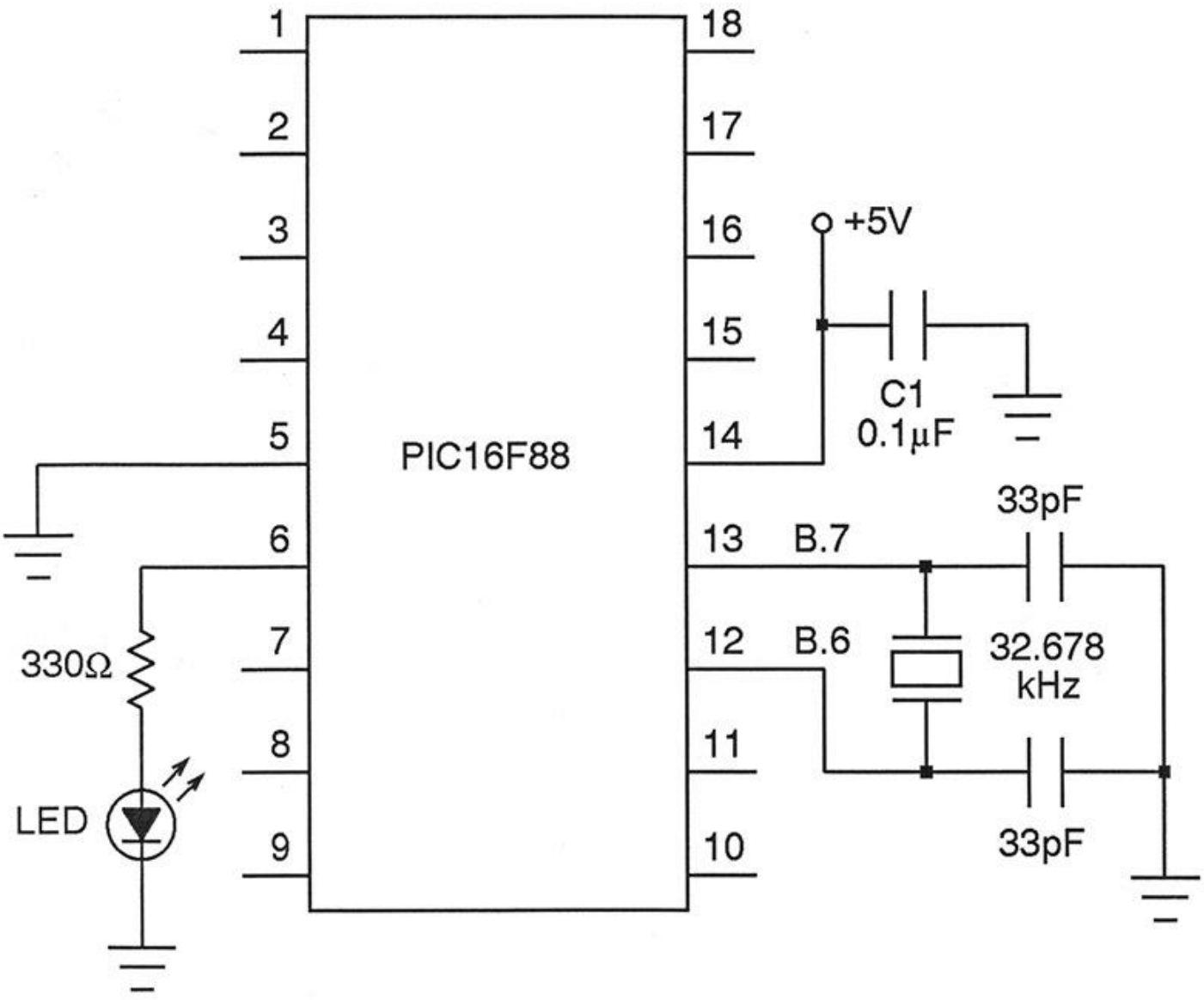

FIGURE 9. O modo T1OSC usa um cristal de relógio como base de tempo.

LISTA DE EXPERIMENTOS

#	Característica Ilustrada	Código Fonte
1	INTRC, A.6 livre	INTRC.GCB
2	INTRC, A.6 gerando φ/4	INTRC.GCB
3	INTOSC, A.6 livre	INTOSC.GCB
4	INTOSC, A.6 gerando φ/4	INTOSC.GCB
5	RCIO, A.6 livre	RC-RCIO.GCB
6	RC, A.6 gerando φ/4	RC-RCIO.GCB
7	LP, cristal de baixa potência de 32.768 kHz	LP.GCB
8	XT, cristal de média potência de 4 MHz	XT.GCB
9	HS, cristal de alta velocidade de 19.6608 MHz	HS.GCB
10	ECIO, CI 555 como fonte de clock externa	ECIO.GCB
11	À prova de falhas, arranque o cristal e veja a INTOSC assumir o controle	FAILSAFE.GCB
12	Mudando de secundário para primário na reinicialização	SWITCHING.GCB
13	T1OSC como um clock secundário	T1OSC.GCB